

# Optimal Receding Horizon Control for Finite Deterministic Systems with Temporal Logic Constraints

Mária Svoreňová, Ivana Černá and Calin Belta

**Abstract**—In this paper, we develop a provably correct optimal control strategy for a finite deterministic transition system. By assuming that penalties with known probabilities of occurrence and dynamics can be sensed locally at the states of the system, we derive a receding horizon strategy that minimizes the expected average cumulative penalty incurred between two consecutive satisfactions of a desired property. At the same time, we guarantee the satisfaction of correctness specifications expressed as Linear Temporal Logic formulas. We illustrate the approach with a persistent surveillance robotics application.

## I. INTRODUCTION

Temporal logics, such as Computation Tree Logic (CTL) and Linear Temporal Logic (LTL), have been customarily used to specify the correctness of computer programs and digital circuits modeled as finite-state transition systems [1]. The problem of analyzing such a model against a temporal logic formula, known as formal analysis or model checking, has received a lot of attention during the past thirty years, and several efficient algorithms and software tools are available [2], [3], [4]. The formal synthesis problem, in which the goal is to design or control a system from a temporal logic specification, has not been studied extensively until a few years ago. Recent results include the use of model checking algorithms for controlling deterministic systems [5], automata games for controlling non-deterministic systems [6], linear programming and value iteration for synthesis of control policies for Markov decision processes [1], [7]. Through the use of abstractions, such techniques have also been used for infinite systems, such as continuous and discrete-time linear systems [8], [9], [10], [11], [12].

The connection between optimal and temporal logic control is an intriguing problem with a potentially high impact in several applications. By combining these two seemingly unrelated areas, our goal is to optimize the behavior of a system subject to correctness constraints. Consider, for example, a mobile robot involved in a persistent surveillance mission in a dangerous area and under tight fuel / time constraints. The correctness requirement is expressed as a temporal logic specification, *e.g.*, “Alternately keep visiting *A* and *B* while always avoiding *C*”, while the resource constraints translate to minimizing a cost function over the feasible trajectories of the robot. While optimal control

is a mature discipline and formal synthesis is fairly well understood, optimal formal synthesis is a largely open area.

In this paper, we focus on finite labeled transition systems and correctness specifications given as formulas of LTL. We assume there is a penalty associated with the states of the system with a known occurrence probability and time-behavior. Motivated by persistent surveillance robotic missions, our goal is to minimize the expected average cumulative penalty incurred between two consecutive satisfactions of a desired property associated with some states of the system, while at the same time satisfying an additional temporal logic constraint. Also from robotics comes our assumption that actual penalty values can only be sensed locally in a close proximity from the current state during the execution of the system. We propose two algorithms for this problem. The first operates offline, *i.e.*, without executions of the system, and therefore only uses the known probabilities but does not exploit actual penalties sensed during the execution. The second algorithm designs an online strategy by locally improving the offline strategy based on local sensing and simulation over a user-defined planning horizon. While both algorithms guarantee optimal expected average penalty collection, in real executions of the system, the second algorithm provides lower real average than the first algorithm. We illustrate these results on a robotic persistent surveillance case study.

This paper is closely related to [13], [14], [5], which also focused on optimal control for finite transitions systems with temporal logic constraints. In [5], the authors developed an offline control strategy minimizing the maximum cost between two consecutive visits to a given set of states, subject to constraints expressed as LTL formulas. Time-varying, locally sensed rewards were introduced in [13], where a receding horizon control strategy maximizing rewards collected locally was shown to satisfy an LTL specification. This approach was generalized in [14] to allow for a broader class of optimization objectives and reward models. In contrast with [13], [14], we interpret the dynamic values appearing in states of the system as penalties instead of rewards, *i.e.*, in our case, the cost function is being minimized rather than maximized. That allows the existence of the optimum in expected average penalty collection. In this paper, we show how it can be achieved using automata-based approach and game theory results.

In Sec. II, we introduce the notation and definitions necessary throughout the paper. The problem is stated in Sec. III. The main results of the paper are in Sec. IV and Sec. V. The simulation results are presented in Sec. VI.

M. Svoreňová, I. Černá are with Faculty of Informatics, Masaryk University, Brno, Czech Republic, svorenova@mail.muni.cz, cerna@muni.cz. C. Belta is with Department of Mechanical Engineering and the Division of Systems Engineering, Boston University, Boston, MA, USA, cbelta@bu.edu. This work was partially supported at MU by grants GAP202/11/0312, LH11065, at BU by ONR grants MURI N00014-09-1051, MURI N00014-10-10952 and by NSF grant CNS-1035588.

## II. PRELIMINARIES

For a set  $S$ , we use  $S^\omega$  and  $S^+$  to denote the set of all infinite and all non-empty finite sequences of elements of  $S$ , respectively. For a finite or infinite sequence  $\alpha = a_0 a_1 \dots$ , we use  $\alpha(i) = a_i$  to denote the  $i$ -th element and  $\alpha^{(i)} = a_0 \dots a_i$  for the finite prefix of  $\alpha$  of length  $|\alpha^{(i)}| = i + 1$ .

**Definition 1:** A weighted deterministic transition system (TS) is a tuple  $\mathcal{T} = (S, T, AP, L, w)$ , where  $S$  is a non-empty finite set of states,  $T \subseteq S \times S$  is a transition relation,  $AP$  is a finite set of atomic propositions,  $L: S \rightarrow 2^{AP}$  is a labeling function and  $w: T \rightarrow \mathbb{N}$  is a weight function. We assume that for every  $s \in S$  exists  $s' \in S$  such that  $(s, s') \in T$ . An initialized transition system is a TS  $\mathcal{T} = (S, T, AP, L, w)$  with a distinctive initial state  $s_{init} \in S$ .

A run of a TS  $\mathcal{T}$  is an infinite sequence  $\rho = s_0 s_1 \dots \in S^\omega$  such that for every  $i \geq 0$  it holds  $(s_i, s_{i+1}) \in T$ . We use  $\text{inf}(\rho)$  to denote the set of all states visited infinitely many times in the run  $\rho$  and  $\text{Run}^{\mathcal{T}}(s)$  for the set of all runs of  $\mathcal{T}$  that start in  $s \in S$ . Let  $\text{Run}^{\mathcal{T}} = \bigcup_{s \in S} \text{Run}^{\mathcal{T}}(s)$ . A finite run  $\sigma = s_0 \dots s_n$  of  $\mathcal{T}$  is a finite prefix of a run of  $\mathcal{T}$  and  $\text{Run}_{\text{fin}}^{\mathcal{T}}(s)$  denotes the set of all finite runs of  $\mathcal{T}$  that start in  $s \in S$ . Let  $\text{Run}_{\text{fin}}^{\mathcal{T}} = \bigcup_{s \in S} \text{Run}_{\text{fin}}^{\mathcal{T}}(s)$ . The length  $|\sigma|$ , or number of stages, of a finite run  $\sigma = s_0 \dots s_n$  is  $n + 1$  and  $\text{last}(\sigma) = s_n$  denotes the last state of  $\sigma$ . With slight abuse of notation, we use  $w(\sigma)$  to denote the weight of a finite run  $\sigma = s_0 \dots s_n$ , i.e.,  $w(\sigma) = \sum_{i=0}^{n-1} w((s_i, s_{i+1}))$ . Moreover,  $w^*(s, s')$  denotes the minimum weight of a finite run from  $s$  to  $s'$ . Specifically,  $w^*(s, s) = 0$  for every  $s \in S$  and if there does not exist a run from  $s$  to  $s'$ , then  $w^*(s, s') = \infty$ . For a set  $S' \subseteq S$  we let  $w^*(s, S') = \min_{s' \in S'} w^*(s, s')$ . We say that a state  $s'$  and a set  $S'$  is reachable from  $s$ , iff  $w^*(s, s') \neq \infty$  and  $w^*(s, S') \neq \infty$ , respectively.

Every run  $\rho = s_0 s_1 \dots \in \text{Run}^{\mathcal{T}}$ , resp.  $\sigma = s_0 \dots s_n \in \text{Run}_{\text{fin}}^{\mathcal{T}}$ , induces a word  $z = L(s_0)L(s_1) \dots \in (2^{AP})^\omega$ , resp.  $z = L(s_0) \dots L(s_n) \in (2^{AP})^+$ , over the power set of  $AP$ .

A cycle of the TS  $\mathcal{T}$  is a finite run  $\text{cyc} = c_0 \dots c_m$  of  $\mathcal{T}$  for which it holds that  $(c_m, c_0) \in T$ .

**Definition 2:** A sub-system of a  $\mathcal{T} = (S, T, AP, L, w)$  is a TS  $\mathcal{U} = (S_{\mathcal{U}}, T_{\mathcal{U}}, AP, L|_{\mathcal{U}}, w|_{\mathcal{U}})$ , where  $S_{\mathcal{U}} \subseteq S$  and  $T_{\mathcal{U}} \subseteq T \cap (S_{\mathcal{U}} \times S_{\mathcal{U}})$ . We use  $L|_{\mathcal{U}}$  to denote the labeling function  $L$  restricted to the set  $S_{\mathcal{U}}$ . Similarly, we use  $w|_{\mathcal{U}}$  with the obvious meaning. If the context is clear, we use  $L, w$  instead of  $L|_{\mathcal{U}}, w|_{\mathcal{U}}$ . A sub-system  $\mathcal{U}$  of  $\mathcal{T}$  is called strongly connected if for every pair of states  $s, s' \in S_{\mathcal{U}}$ , there exists a finite run  $\sigma \in \text{Run}_{\text{fin}}^{\mathcal{U}}(s)$  such that  $\text{last}(\sigma) = s'$ . A strongly connected component (SCC) of  $\mathcal{T}$  is a maximal strongly connected sub-system of  $\mathcal{T}$ . We use  $\text{SCC}(\mathcal{T})$  to denote the set of all strongly connected components of  $\mathcal{T}$ .

Strongly connected components of a TS  $\mathcal{T}$  are pairwise disjoint. Hence, the cardinality of the set  $\text{SCC}(\mathcal{T})$  is bounded by the number of states of  $\mathcal{T}$  and the set can be computed using Tarjan's algorithm [15].

**Definition 3:** Let  $\mathcal{T} = (S, T, AP, L, w)$  be a TS. A control strategy for  $\mathcal{T}$  is a function  $C: \text{Run}_{\text{fin}}^{\mathcal{T}} \rightarrow S$  such that for every  $\sigma \in \text{Run}_{\text{fin}}^{\mathcal{T}}$ , it holds that  $(\text{last}(\sigma), C(\sigma)) \in T$ .

A strategy  $C$  for which  $C(\sigma_1) = C(\sigma_2)$ , for all finite runs  $\sigma_1, \sigma_2 \in \text{Run}_{\text{fin}}^{\mathcal{T}}$  with  $\text{last}(\sigma_1) = \text{last}(\sigma_2)$ , is called memoryless. In that case,  $C$  is a function  $C: S \rightarrow S$ .

A strategy is called finite-memory if it is defined as a tuple  $C = (M, \text{next}, \Delta, \text{start})$ , where  $M$  is a finite set of modes,  $\Delta: M \times S \rightarrow M$  is a transition function,  $\text{next}: M \times S \rightarrow S$  selects a state of  $\mathcal{T}$  to be visited next, and  $\text{start}: S \rightarrow M$  selects the starting mode for every  $s \in S$ .

A run induced by a strategy  $C$  for  $\mathcal{T}$  is a run  $\rho_C = s_0 s_1 \dots \in \text{Run}^{\mathcal{T}}$  for which  $s_{i+1} = C(\rho_C^{(i)})$  for every  $i \geq 0$ . For every  $s \in S$ , there is exactly one run induced by  $C$  that starts in  $s$ . A finite run induced by  $C$  is  $\sigma_C \in \text{Run}_{\text{fin}}^{\mathcal{T}}$ , which is a finite prefix of a run  $\rho_C$  induced by  $C$ .

Let  $C$  be a strategy, finite-memory or not, for a TS  $\mathcal{T}$ . For every state  $s \in S$ , the run  $\rho_C \in \text{Run}^{\mathcal{T}}(s)$  induced by  $C$  satisfies  $\text{inf}(\rho_C) \subseteq S_{\mathcal{U}}$  for some  $\mathcal{U} \in \text{SCC}(\mathcal{T})$  [1]. We say that  $C$  leads  $\mathcal{T}$  from the state  $s$  to the SCC  $\mathcal{U}$ .

**Definition 4:** Linear Temporal Logic (LTL) formulas over the set  $AP$  are formed according to the following grammar:  $\phi ::= \text{true} \mid a \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathbf{X}\phi \mid \phi \mathbf{U}\phi \mid \mathbf{G}\phi \mid \mathbf{F}\phi$ , where  $a \in AP$  is an atomic proposition,  $\neg, \vee$  and  $\wedge$  are standard Boolean connectives, and  $\mathbf{X}$  (*next*),  $\mathbf{U}$  (*until*),  $\mathbf{G}$  (*always*) and  $\mathbf{F}$  (*eventually*) are temporal operators.

The semantics of LTL is defined over words over  $2^{AP}$ , such as those generated by the runs of a TS  $\mathcal{T}$  (for details see e.g., [1]). For example, a word  $w \in (2^{AP})^\omega$  satisfies  $\mathbf{G}\phi$  and  $\mathbf{F}\phi$  if  $\phi$  holds in  $w$  always and eventually, respectively. If the word induced by a run of  $\mathcal{T}$  satisfies a formula  $\phi$ , we say that the run satisfies  $\phi$ . We call  $\phi$  satisfiable in  $\mathcal{T}$  from  $s \in S$  if there exists a run  $\rho \in \text{Run}^{\mathcal{T}}(s)$  that satisfies  $\phi$ .

Having an initialized TS  $\mathcal{T}$  and an LTL formula  $\phi$  over  $AP$ , the formal synthesis problem aims to find a strategy  $C$  for  $\mathcal{T}$  such that the run  $\rho_C \in \text{Run}^{\mathcal{T}}(s_{init})$  induced by  $C$  satisfies  $\phi$ . In that case we also say that the strategy  $C$  satisfies  $\phi$ . The formal synthesis problem can be solved using principles from model checking methods [1]. Specifically,  $\phi$  is translated to a Büchi automaton and the system combining the Büchi automaton and the TS  $\mathcal{T}$  is analyzed.

**Definition 5:** A Büchi automaton (BA) is a tuple  $\mathcal{B} = (Q, 2^{AP}, \delta, q_0, F)$ , where  $Q$  is a non-empty finite set of states,  $2^{AP}$  is the alphabet,  $\delta \subseteq Q \times 2^{AP} \times Q$  is a transition relation such that for every  $q \in Q$ ,  $a \in 2^{AP}$ , there exists  $q' \in Q$  such that  $(q, a, q') \in \delta$ ,  $q_0 \in Q$  is the initial state, and  $F \subseteq Q$  is a set of accepting states.

A run  $q_0 q_1 \dots \in Q^\omega$  of  $\mathcal{B}$  is an infinite sequence such that for every  $i \geq 0$  there exists  $a_i \in 2^{AP}$  with  $(q_i, a_i, q_{i+1}) \in \delta$ . The word  $a_0 a_1 \dots \in (2^{AP})^\omega$  is called the word induced by the run  $q_0 q_1 \dots$ . A run  $q_0 q_1 \dots$  of  $\mathcal{B}$  is accepting if there exist infinitely many  $i \geq 0$  such that  $q_i$  is an accepting state.

For every LTL formula  $\phi$  over  $AP$ , one can construct a Büchi automaton  $\mathcal{B}_\phi$  such that the accepting runs are all and only words over  $2^{AP}$  satisfying  $\phi$  [16]. We refer readers to [17], [18] for algorithms and to online implementations such as [19], to translate an LTL formula to a BA.

**Definition 6:** Let  $\mathcal{T} = (S, T, AP, L, w)$  be an initialized TS and  $\mathcal{B} = (Q, 2^{AP}, \delta, q_0, F)$  be a Büchi automaton. The product  $\mathcal{P}$  of  $\mathcal{T}$  and  $\mathcal{B}$  is a tuple  $\mathcal{P} =$

$(S_{\mathcal{P}}, T_{\mathcal{P}}, s_{\mathcal{P}init}, AP, L_{\mathcal{P}}, F_{\mathcal{P}}, w_{\mathcal{P}})$ , where  $S_{\mathcal{P}} = S \times Q$ ,  $T_{\mathcal{P}} \subseteq S_{\mathcal{P}} \times S_{\mathcal{P}}$  is a transition relation such that for every  $(s, q), (s', q') \in S_{\mathcal{P}}$  it holds that  $((s, q), (s', q')) \in T_{\mathcal{P}}$  if and only if  $(s, s') \in T$  and  $(q, L(s), q') \in \delta$ ,  $s_{\mathcal{P}init} = (s_{init}, q_0)$  is the initial state,  $L_{\mathcal{P}}((s, q)) = L(s)$  is a labeling function,  $F_{\mathcal{P}} = S \times F$  is a set of accepting states, and  $w_{\mathcal{P}}(((s, q), (s', q')))) = w((s, s'))$  is a weight function.

The product  $\mathcal{P}$  can be viewed as an initialized TS with a set of accepting states. Therefore, we adopt the definitions of a run  $\rho$ , a finite run  $\sigma$ , its weight  $w_{\mathcal{P}}(\sigma)$ , and sets  $\text{Run}^{\mathcal{P}}((s, q))$ ,  $\text{Run}^{\mathcal{P}}$ ,  $\text{Run}_{\text{fin}}^{\mathcal{P}}((s, q))$  and  $\text{Run}_{\text{fin}}^{\mathcal{P}}$  from above. Similarly, a cycle  $\text{cyc}$  of  $\mathcal{P}$ , a strategy  $C_{\mathcal{P}}$  for  $\mathcal{P}$  and runs  $\rho_{C_{\mathcal{P}}}, \sigma_{C_{\mathcal{P}}}$  induced by  $C_{\mathcal{P}}$  are defined in the same way as for a TS. On the other hand,  $\mathcal{P}$  can be viewed as a weighted BA over the trivial alphabet with a labeling function, which gives us the definition of an accepting run of  $\mathcal{P}$ .

Using the projection on the first component, every run  $(s_0, q_0)(s_1, q_1) \dots$  and finite run  $(s_0, q_0) \dots (s_n, q_n)$  of  $\mathcal{P}$  corresponds to a run  $s_0 s_1 \dots$  and a finite run  $s_0 \dots s_n$  of  $\mathcal{T}$ , respectively. Vice versa, for every run  $s_0 s_1 \dots$  and finite run  $s_0 \dots s_n$  of  $\mathcal{T}$ , there exists a run  $(s_0, q_0)(s_1, q_1) \dots$  and finite run  $(s_0, q_0) \dots (s_n, q_n)$ . Similarly, every strategy for  $\mathcal{P}$  projects to a strategy for  $\mathcal{T}$  and for every strategy for  $\mathcal{T}$  there exists a strategy for  $\mathcal{P}$  that projects to it. The projection of a finite-memory strategy for  $\mathcal{P}$  is also finite-memory.

Since  $\mathcal{P}$  can be viewed as a TS, we also adopt the definitions of a sub-system and a strongly connected component.

**Definition 7:** Let  $\mathcal{P} = (S_{\mathcal{P}}, T_{\mathcal{P}}, s_{\mathcal{P}init}, AP, L_{\mathcal{P}}, F_{\mathcal{P}}, w_{\mathcal{P}})$  be the product of an initialized TS  $\mathcal{T}$  and a BA  $\mathcal{B}$ . An accepting strongly connected component (ASCC) of  $\mathcal{P}$  is an SCC  $\mathcal{U} = (S_{\mathcal{U}}, T_{\mathcal{U}}, AP, L_{\mathcal{P}}, w_{\mathcal{P}})$  such that the set  $S_{\mathcal{U}} \cap F_{\mathcal{P}}$  is non-empty and we refer to it as the set  $F_{\mathcal{U}}$  of accepting states of  $\mathcal{U}$ . We use  $\text{ASCC}(\mathcal{P})$  to denote the set of all ASCCs of  $\mathcal{P}$  that are reachable from the initial state  $s_{\mathcal{P}init}$ .

In our work, we always assume that  $\text{ASCC}(\mathcal{P})$  is non-empty, i.e., the given LTL formula is satisfiable in the TS.

### III. PROBLEM FORMULATION

Consider an initialized weighted transition system  $\mathcal{T} = (S, T, AP, L, w)$ . The weight  $w((s, s'))$  of a transition  $(s, s') \in T$  represents the amount of time that the transition takes and the system starts at time 0. We use  $t_n$  to denote the point in time after the  $n$ -th transition of a run, i.e., initially the system is in a state  $s_0$  at time  $t_0 = 0$  and after a finite run  $\sigma \in \text{Run}_{\text{fin}}^{\mathcal{T}}(s_0)$  of length  $n + 1$  the time is  $t_n = w(\sigma)$ .

We assume there is a dynamic *penalty* associated with every state  $s \in S$ . In this paper, we address the following model of penalties. Nevertheless, as we discuss in Sec.V, the algorithms presented in the next section provide optimal solution for a much broader class of penalty dynamics. The penalty is a rational number between 0 and 1 that is increasing every time unit by  $\frac{1}{r}$ , where  $r \in \mathbb{N}$  is a given rate. Always when the penalty is 1, in the next time unit the penalty remains 1 or it drops to 0 according to a given probability distribution. Upon the visit of a state, the corresponding penalty is incurred. We assume that the visit of the state does not affect the penalty's value or dynamics.

Formally, the penalties are defined by a *rate*  $r \in \mathbb{N}$  and a *penalty probability function*  $p: S \rightarrow (0, 1]$ , where  $p(s)$  is the probability that if the penalty in a state  $s$  is 1 then in the next time unit the penalty remains 1, and  $1 - p(s)$  is the probability of the penalty dropping to 0. The penalties are described using a function  $g: S \times \mathbb{N}_0 \rightarrow \{\frac{i}{r} \mid i \in \{0, 1, \dots, r\}\}$ , such that  $g(s, t)$  is the penalty in a state  $s \in S$  at time  $t \in \mathbb{N}_0$ . For  $s \in S$ ,  $g(s, 0)$  is a uniformly distributed random variable with values in the set  $\{\frac{i}{r} \mid i \in \{0, 1, \dots, r\}\}$  and for  $t \geq 1$

$$g(s, t) = \begin{cases} g(s, t-1) + \frac{1}{r} & \text{if } g(s, t-1) < 1, \\ x & \text{otherwise,} \end{cases} \quad (1)$$

where  $x$  is a random variable such that  $x = 1$  with probability  $p(s)$  and  $x = 0$  otherwise. We use

$$g_{\text{exp}}(s) = (1 - p(s)) \cdot \frac{1}{2} + p(s) \cdot 1 = \frac{1}{2}(1 + p(s)) \quad (2)$$

to denote the expected value of the penalty in a state  $s \in S$ . Please note that  $\frac{1}{2} \leq g_{\text{exp}}(s) \leq 1$ , for every  $s \in S$ .

In our setting, the penalties are sensed only locally in the states in close proximity from the current state. To be specific, we assume a *visibility range*  $v \in \mathbb{N}$  is given. If the system is in a state  $s \in S$  at time  $t$ , the penalty  $g(s', t)$  of a state  $s' \in S$  is observable if and only if  $s' \in \text{Vis}(s) = \{s' \in S \mid w^*(s, s') \leq v\}$ . The set  $\text{Vis}(s)$  is also called the set of states visible from  $s$ .

The problem we consider in this paper combines the formal synthesis problem with long-term optimization of the expected amount of penalties incurred during the system's execution. We assume that the specification is given as an LTL formula  $\phi$  of the form

$$\phi = \varphi \wedge \mathbf{GF}\pi_{\text{sur}}, \quad (3)$$

where  $\varphi$  is an LTL formula over  $AP$  and  $\pi_{\text{sur}} \in AP$ . This formula requires that the system satisfies  $\varphi$  and surveys the states satisfying the property  $\pi_{\text{sur}}$  infinitely often. We say that every visit of a state from the set  $S_{\text{sur}} = \{s \in S \mid \pi_{\text{sur}} \in L(s)\}$  completes a surveillance cycle. Specifically, starting from the initial state, the first visit of  $S_{\text{sur}}$  after the initial state completes the first surveillance cycle of a run. Note that a surveillance cycle is not a cycle in the sense of the definition of a cycle of a TS in Sec. II. For a finite run  $\sigma$  such that  $\text{last}(\sigma) \in S_{\text{sur}}$ ,  $\sharp(\sigma)$  denotes the number of complete surveillance cycles in  $\sigma$ , otherwise  $\sharp(\sigma)$  is the number of complete surveillance cycles plus one. We define a function  $V_{\mathcal{T}, C}: S \rightarrow \mathbb{R}_0^+$  such that  $V_{\mathcal{T}, C}(s)$  is the expected average cumulative penalty per surveillance cycle (APPC) incurred under a strategy  $C$  for  $\mathcal{T}$  starting from a state  $s \in S$ :

$$V_{\mathcal{T}, C}(s) = \limsup_{n \rightarrow \infty} E\left(\frac{\sum_{i=0}^n g(\rho_C(i), w(\rho_C(i)))}{\sharp(\rho_C^{(n)})}\right), \quad (4)$$

where  $\rho_C \in \text{Run}^{\mathcal{T}}(s)$  is the run induced by  $C$  starting from  $s$  and  $E(\cdot)$  denotes the expected value. In this paper, we consider the following problem:

**Problem 1:** Let  $\mathcal{T} = (S, T, AP, L, w)$  be an initialized TS, with penalties defined by a rate  $r \in \mathbb{N}$  and penalty probabilities  $p: S \rightarrow (0, 1]$ . Let  $v \in \mathbb{N}$  be a visibility range and  $\phi$  an LTL formula over the set  $AP$  of the form in Eq. (3). Find a strategy  $C$  for  $\mathcal{T}$  such that  $C$  satisfies  $\phi$  and

among all strategies satisfying  $\phi$ ,  $C$  minimizes the APPC value  $V_{\mathcal{T},C}(s_{init})$  defined in Eq. (4).

In the next section, we propose two algorithms solving the above problem. The first algorithm operates offline, without the deployment of the system, and therefore, without taking advantage of the local sensing of penalties. On the other hand, the second algorithm computes the strategy in real-time by locally improving the offline strategy according to the penalties observed from the current state and their simulation over the next  $h$  time units, where  $h \geq 1$  is a natural number, a user-defined *planning horizon*.

#### IV. SOLUTION

The two algorithms work with the product  $\mathcal{P} = (S_{\mathcal{P}}, T_{\mathcal{P}}, s_{\mathcal{P}init}, AP, L_{\mathcal{P}}, F_{\mathcal{P}}, w_{\mathcal{P}})$  of the initialized TS  $\mathcal{T}$  and a Büchi automaton  $\mathcal{B}_{\phi}$  for the LTL formula  $\phi$ . To project the penalties from  $\mathcal{T}$  to  $\mathcal{P}$ , we define the penalty in a state  $(s, q) \in S_{\mathcal{P}}$  at time  $t$  as  $g((s, q), t) = g(s, t)$ . We also adopt the visibility range  $v$  and the set  $\text{Vis}((s, q))$  of all states visible from  $(s, q)$  is defined as for a state of  $\mathcal{T}$ . The APPC function  $V_{\mathcal{P},C_{\mathcal{P}}}$  of a strategy  $C_{\mathcal{P}}$  for  $\mathcal{P}$  is then defined according to Eq. (4). We use the correspondence between the strategies for  $\mathcal{P}$  and  $\mathcal{T}$  to find a strategy for  $\mathcal{T}$  that solves Problem 1. Let  $C_{\mathcal{P}}$  be a strategy for  $\mathcal{P}$  such that the run induced by  $C_{\mathcal{P}}$  visits the set  $F_{\mathcal{P}}$  infinitely many times and at the same time, the APPC value  $V_{\mathcal{P},C_{\mathcal{P}}}(s_{\mathcal{P}init})$  is minimal among all strategies that visit  $F_{\mathcal{P}}$  infinitely many times. It is easy to see that  $C_{\mathcal{P}}$  projects to a strategy  $C$  for  $\mathcal{T}$  that solves Problem 1 and  $V_{\mathcal{T},C}(s_{init}) = V_{\mathcal{P},C_{\mathcal{P}}}(s_{\mathcal{P}init})$ .

The offline algorithm leverages ideas from formal methods. Using the automata-based approach to model checking, one can construct a strategy  $C_{\mathcal{P}}^{\phi}$  for  $\mathcal{P}$  that visits at least one of the accepting states infinitely many times. On the other hand, using graph theory, we can design a strategy  $C_{\mathcal{P}}^V$  that achieves the minimum APPC value among all strategies of  $\mathcal{P}$  that do not cause an immediate, unrepairable violation of  $\phi$ , *i.e.*,  $\phi$  is satisfiable from every state of the run induced by  $C_{\mathcal{P}}^V$ . However, we would like to have a strategy  $C_{\mathcal{P}}$  satisfying both properties at the same time. To achieve that, we employ a technique from game theory presented in [20]. Intuitively, we combine two strategies  $C_{\mathcal{P}}^{\phi}$  and  $C_{\mathcal{P}}^V$  to create a new strategy  $C_{\mathcal{P}}$ . The strategy  $C_{\mathcal{P}}$  is played in rounds, where each round consists of two phases. In the first phase, we play the strategy  $C_{\mathcal{P}}^{\phi}$  until an accepting state is reached. We say that the system is to achieve the mission subgoal. The second phase applies the strategy  $C_{\mathcal{P}}^V$ . The aim is to maintain the expected average cumulative penalty per surveillance cycle in the current round, and we refer to it as the average subgoal. The number of steps for which we apply  $C_{\mathcal{P}}^V$  is computed individually every time we enter the second phase of a round.

The online algorithm constructs a strategy  $C_{\mathcal{P}}$  by locally improving the strategy  $C_{\mathcal{P}}$  computed by the offline algorithm. Intuitively, we compare applying  $C_{\mathcal{P}}$  for several steps to reach a specific state or set of states of  $\mathcal{P}$ , to executing different local paths to reach the same state or set. We consider a finite set of finite runs leading to the state, or

set, containing the finite run induced by  $C_{\mathcal{P}}$ , choose the one that is expected to minimize the average cumulative penalty per surveillance cycle incurred in the current round and apply the first transition of the chosen run. The process continues until the state, or set, is reached, and then it starts over again.

##### A. Probability measure

Let  $C_{\mathcal{P}}$  be a strategy for  $\mathcal{P}$  and  $(s, q) \in S_{\mathcal{P}}$  a state of  $\mathcal{P}$ . For a finite run  $\sigma_{C_{\mathcal{P}}} \in \text{Run}_{\text{fin}}^{\mathcal{P}}((s, q))$  induced by the strategy  $C_{\mathcal{P}}$  starting from the state  $(s, q)$  and a sequence  $\tau \in (\{\frac{i}{r} \mid 0 \leq i \leq r\})^+$  of length  $|\sigma_{C_{\mathcal{P}}}|$ , we call  $(\sigma_{C_{\mathcal{P}}}, \tau)$  a finite pair. Analogously, an infinite pair  $(\rho_{C_{\mathcal{P}}}, \kappa)$  consists of the run  $\rho_{C_{\mathcal{P}}} \in \text{Run}^{\mathcal{P}}((s, q))$  induced by the strategy  $C_{\mathcal{P}}$  and an infinite sequence  $\kappa \in (\{\frac{i}{r} \mid 0 \leq i \leq r\})^{\omega}$ . A cylinder set  $\text{Cyl}((\sigma_{C_{\mathcal{P}}}, \tau))$  of a finite pair  $(\sigma_{C_{\mathcal{P}}}, \tau)$  is the set of all infinite pairs  $(\rho_{C_{\mathcal{P}}}, \kappa)$  such that  $\tau$  is a prefix of  $\kappa$ .

Consider the  $\sigma$ -algebra generated by the set of cylinder sets of all finite pairs  $(\sigma_{C_{\mathcal{P}}}, \tau)$ , where  $\sigma_{C_{\mathcal{P}}} \in \text{Run}_{\text{fin}}^{\mathcal{P}}((s, q))$  is a finite run induced by the strategy  $C_{\mathcal{P}}$  starting from the state  $(s, q)$  and  $\tau \in (\{\frac{i}{r} \mid 0 \leq i \leq r\})^+$  is of length  $|\sigma_{C_{\mathcal{P}}}|$ . From classical concepts in probability theory [21], there exists a unique probability measure  $\text{Pr}_{(s,q)}^{\mathcal{P},C_{\mathcal{P}}}$  on the  $\sigma$ -algebra such that for a finite pair  $(\sigma_{C_{\mathcal{P}}}, \tau)$

$$\text{Pr}_{(s,q)}^{\mathcal{P},C_{\mathcal{P}}}(\text{Cyl}((\sigma_{C_{\mathcal{P}}}, \tau)))$$

is the probability that the penalties incurred in the first  $|\sigma_{C_{\mathcal{P}}}| + 1$  stages when applying the strategy  $C_{\mathcal{P}}$  in  $\mathcal{P}$  from the state  $(s, q)$  are given by the sequence  $\tau$ , *i.e.*,

$$g(\sigma_{C_{\mathcal{P}}}(i), w_{\mathcal{P}}(\sigma_{C_{\mathcal{P}}}^i)) = \tau(i)$$

for every  $0 \leq i \leq |\sigma_{C_{\mathcal{P}}}|$ . This probability is given by the penalty dynamics and therefore, can be computed from the rate  $r$  and the penalty probability function  $p$ . For a set  $X$  of infinite pairs, an element of the above  $\sigma$ -algebra, the probability  $\text{Pr}_{(s,q)}^{\mathcal{P},C_{\mathcal{P}}}(X)$  is the probability that under  $C_{\mathcal{P}}$  starting from  $(s, q)$  the infinite sequence of penalties received in the visited states is  $\kappa$  where  $(\rho_{C_{\mathcal{P}}}, \kappa) \in X$ .

##### B. Offline control

In this section, we construct a strategy  $C_{\mathcal{P}}$  for  $\mathcal{P}$  that projects to a strategy  $C$  for  $\mathcal{T}$  solving Problem 1. The strategy  $C_{\mathcal{P}}$  has to visit  $F_{\mathcal{P}}$  infinitely many times and therefore,  $C_{\mathcal{P}}$  must lead from  $s_{\mathcal{P}init}$  to an ASCC. For an  $\mathcal{U} \in \text{ASCC}(\mathcal{P})$ , we denote  $V_{\mathcal{U}}^*((s, q))$  the minimum expected average cumulative penalty per surveillance cycle that can be achieved in  $\mathcal{U}$  starting from  $(s, q) \in S_{\mathcal{U}}$ . Since  $\mathcal{U}$  is strongly connected, this value is the same for all the states in  $S_{\mathcal{U}}$  and is referred to as  $V_{\mathcal{U}}^*$ . It is associated with a cycle  $\text{cyc}_{\mathcal{U}}^V = c_0 \dots c_m$  of  $\mathcal{U}$  witnessing the value, *i.e.*,

$$\frac{1}{|\text{cyc}_{\mathcal{U}}^V \cap S_{\mathcal{U} \text{sur}}|} \sum_{i=0}^m g_{\text{exp}}(c_i) = V_{\mathcal{U}}^*$$

where  $S_{\mathcal{U} \text{sur}}$  is the set of all states of  $\mathcal{U}$  labeled with  $\pi_{\text{sur}}$ . Since  $\mathcal{U}$  is an ASCC, it holds  $S_{\mathcal{U} \text{sur}} \neq \emptyset$ .

We design an algorithm that finds the value  $V_{\mathcal{U}}^*$  and a cycle  $\text{cyc}_{\mathcal{U}}^V$  for an ASCC  $\mathcal{U}$ . The algorithm first reduces  $\mathcal{U}$  to a TS  $\bar{\mathcal{U}}$  and then applies the Karp's algorithm [22] on  $\bar{\mathcal{U}}$ . The Karp's algorithm finds for a directed graph with values

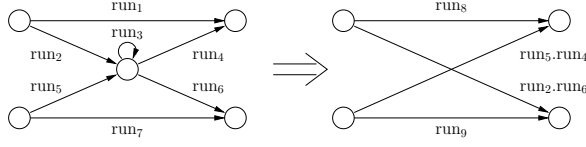


Fig. 1: An example of elimination of a state during the reduction of an ASCC  $\mathcal{U}$ . The finite run  $\text{run}_8$  is equal to the one of the finite runs  $\text{run}_1$  and  $\text{run}_2.\text{run}_4$  that minimizes the sum of expected penalties in the states of the run. Similarly,  $\text{run}_9$  is one of the finite runs  $\text{run}_7$  and  $\text{run}_5.\text{run}_6$ .

on edges a cycle with minimum value per edge also called the minimum mean cycle. The value  $V_{\mathcal{U}}^*$  and cycle  $\text{cyc}_{\mathcal{U}}^V$  are synthesized from the minimum mean cycle.

Let  $\mathcal{U} = (S_{\mathcal{U}}, T_{\mathcal{U}}, AP, L_{\mathcal{P}}, w_{\mathcal{P}})$  be an ASCC of  $\mathcal{P}$ . For simplicity, we use singletons such as  $u, u_i$  to denote the states of  $\mathcal{P}$  in this paragraph. We construct a TS

$$\bar{\mathcal{U}} = (S_{\bar{\mathcal{U}}}, T_{\bar{\mathcal{U}}}, AP, L_{\mathcal{P}}, w_{\bar{\mathcal{U}}}),$$

and a function  $\text{run}: T_{\bar{\mathcal{U}}} \rightarrow \text{Run}_{\text{fin}}^{\mathcal{U}}$  for which it holds that  $(u, u') \in T_{\bar{\mathcal{U}}}$  if and only if there exists a finite run in  $\mathcal{U}$  from  $u \in S_{\bar{\mathcal{U}}}$  to  $u' \in S_{\bar{\mathcal{U}}}$  with one surveillance cycle, i.e., between  $u$  and  $u'$  no state labeled with  $\pi_{\text{sur}}$  is visited. Moreover, the run  $\text{run}((u, u')) = u_0 \dots u_n$  is such that  $u = u_0$  and  $\sigma = u_0 \dots u_n u'$  is the finite run in  $\mathcal{U}$  from  $u$  to  $u'$  with one surveillance cycle that minimizes the expected sum of penalties received during  $\sigma$  among all finite runs in  $\mathcal{U}$  from  $u$  to  $u'$  with one surveillance cycle. The TS can be constructed from  $\mathcal{U}$  by an iterative algorithm eliminating the states from  $S_{\mathcal{U}} \setminus S_{\bar{\mathcal{U}}}$  one by one, in arbitrary order. At the beginning let  $\bar{\mathcal{U}} = \mathcal{U}$ ,  $T_{\bar{\mathcal{U}}} = T_{\mathcal{U}}$ , and for every transition  $(u, u') \in T_{\bar{\mathcal{U}}}$  let  $\text{run}((u, u')) = u$ . The procedure for eliminating  $u \in S_{\mathcal{U}} \setminus S_{\bar{\mathcal{U}}}$  proceeds as follows. Consider every  $u_1 \neq u, u_2 \neq u$  such that  $(u_1, u), (u, u_2) \in T_{\bar{\mathcal{U}}}$ . If the transition  $(u_1, u_2)$  is not in  $T_{\bar{\mathcal{U}}}$ , add  $(u_1, u_2)$  to  $T_{\bar{\mathcal{U}}}$  and define  $\text{run}((u_1, u_2)) = \text{run}((u_1, u)).\text{run}((u, u_2))$ , where  $\cdot$  is the concatenation of sequences. If  $T_{\bar{\mathcal{U}}}$  already contains the transition  $(u_1, u_2)$  and  $\text{run}((u_1, u_2)) = \sigma$ , we set  $\text{run}((u_1, u_2)) = \text{run}((u_1, u)).\text{run}((u, u_2))$ , if

$$\sum g_{\text{exp}}(\text{run}((u_1, u)).\text{run}((u, u_2))) \leq \sum g_{\text{exp}}(\sigma),$$

where  $\sum g_{\text{exp}}(x)$  for a run  $x$  is the sum of  $g_{\text{exp}}(x(i))$  for every state  $x(i)$  of  $x$ , otherwise we let  $\text{run}((u_1, u_2)) = \sigma$ . The weight  $w_{\bar{\mathcal{U}}}((u_1, u_2)) = \sum g_{\text{exp}}(\text{run}((u_1, u_2)))$ . Once all pairs  $u_1, u_2$  are handled, remove  $u$  from  $S_{\bar{\mathcal{U}}}$  and all adjacent transitions from  $T_{\bar{\mathcal{U}}}$ . Fig. 1 demonstrates one iteration of the algorithm.

Consequently, we apply the Karp's algorithm on the oriented graph with vertices  $S_{\bar{\mathcal{U}}}$ , edges  $T_{\bar{\mathcal{U}}}$  and values on edges  $w_{\bar{\mathcal{U}}}$ . Let  $\text{cyc}_{\bar{\mathcal{U}}}^V = u_0 \dots u_m$  be the minimum mean cycle of this graph. Then it holds

$$V_{\bar{\mathcal{U}}}^* = \frac{1}{m+1} \sum_{i=0}^m g_{\text{exp}}(\text{run}((u_i, u_{i+1 \bmod (m+1)}))),$$

$$\text{cyc}_{\bar{\mathcal{U}}}^V = \text{run}((u_0, u_1)). \dots \text{run}((u_{m-1}, u_m)).\text{run}((u_m, u_0)).$$

When the APPC value and the corresponding cycle is computed for every ASCC of  $\mathcal{P}$ , we choose the ASCC that minimizes the APPC value. We denote this ASCC  $\mathcal{U} = (S_{\mathcal{U}}, T_{\mathcal{U}}, AP, L_{\mathcal{P}}, w_{\mathcal{P}})$  and  $\text{cyc}_{\mathcal{U}}^V = c_0 \dots c_m$ .

The mission subgoal aims to reach an accepting state from the set  $F_{\mathcal{U}}$ . The corresponding strategy  $C_{\mathcal{P}}^{\phi}$  is such that from every state  $(s, q) \in S_{\mathcal{P}} \setminus F_{\mathcal{U}}$  that can reach the set  $F_{\mathcal{U}}$ , we follow one of the finite runs with minimum weight from  $(s, q)$  to  $F_{\mathcal{U}}$ . That means,  $C_{\mathcal{P}}^{\phi}$  is a memoryless strategy such that for  $(s, q) \in S_{\mathcal{P}} \setminus F_{\mathcal{U}}$  with  $w_{\mathcal{P}}^{\phi}((s, q), F_{\mathcal{U}}) < \infty$  it holds  $C_{\mathcal{P}}^{\phi}((s, q)) = (s', q')$  where

$$w_{\mathcal{P}}((s, q), (s', q')) = w_{\mathcal{P}}^*((s, q), F_{\mathcal{U}}) - w_{\mathcal{P}}^*((s', q'), F_{\mathcal{U}}).$$

The strategy  $C_{\mathcal{P}}^V$  for the average subgoal is given by the cycle  $\text{cyc}_{\mathcal{U}}^V = c_0 \dots c_m$  of the ASCC  $\mathcal{U}$ . Similarly to the mission subgoal, for a state  $(s, q) \in S_{\mathcal{P}} \setminus \text{cyc}_{\mathcal{U}}^V$  with  $w_{\mathcal{P}}^*((s, q), \text{cyc}_{\mathcal{U}}^V) < \infty$ , the strategy  $C_{\mathcal{P}}^V$  follows one of the finite runs with minimum weight to  $\text{cyc}_{\mathcal{U}}^V$ . For a state  $c_i \in \text{cyc}_{\mathcal{U}}^V$ , it holds  $C_{\mathcal{P}}^V(c_i) = c_{i+1 \bmod (m+1)}$ . If all the states of the cycle  $\text{cyc}_{\mathcal{U}}^V$  are distinct, the strategy  $C_{\mathcal{P}}^V$  is memoryless, otherwise it is finite-memory.

*Proposition 1:* For the strategy  $C_{\mathcal{P}}^V$  and every state  $(s, q) \in S_{\mathcal{U}}$ , it holds

$$\lim_{n \rightarrow \infty} \text{Pr}_{(s, q)}^{U, C_{\mathcal{P}}^V} \left( \frac{\sum_{i=0}^n g(\rho_{C_{\mathcal{P}}^V}(i), w_{\mathcal{P}}(\rho_{C_{\mathcal{P}}^V}^{(i)}))}{\#(\rho_{C_{\mathcal{P}}^V}^{(n)})} \leq V_{\mathcal{U}}^* \right) = 1.$$

Equivalently, for every state  $(s, q) \in S_{\mathcal{U}}$  and every  $\epsilon > 0$ , there exists  $j(\epsilon) \in \mathbb{N}$  such that if the strategy  $C_{\mathcal{P}}^V$  is played from the state  $(s, q)$  until at least  $l \geq j(\epsilon)$  surveillance cycles are completed, then the average cumulative penalty per surveillance cycle incurred in the performed finite run is at most  $V_{\mathcal{U}}^* + \epsilon$  with probability at least  $1 - \epsilon$ .

*Proof:* (Sketch.) The proof is based on the fact that the product  $\mathcal{P}$  with dynamic penalties can be translated into a Markov decision process (MDP) (see e.g., [23]) with static penalties. The run  $\rho_{C_{\mathcal{P}}^V}$  corresponds to a Markov chain (see e.g., [24]) of the MDP. Moreover, the cycle  $\text{cyc}_{\mathcal{U}}^V$  corresponds to the minimum mean cycle of the reduced TS  $\bar{\mathcal{U}}$ . Hence, the equation in the theorem is equivalent to the property of MDPs with static penalties proved in [20] regarding the minimum expected penalty incurred per stage. ■

*Remark 1:* Assume there exists a state  $(s, q) \in S_{\mathcal{P}}$  with  $p((s, q)) = 0$ , i.e., if the penalty in  $(s, q)$  is 1, it always drops to 0. The dynamics of the penalty in  $(s, q)$  is not probabilistic and if we visit  $(s, q)$  infinitely many times, the expected average penalty incurred in  $(s, q)$  might differ from  $g_{\text{exp}}((s, q))$ . That can cause violation of Prop. 1.

Now we describe the strategy  $C_{\mathcal{P}}$ . It is played in rounds, where each round consists of two phases, one for each subgoal. The first round starts at the beginning of the execution of the system in the initial state  $s_{\mathcal{P} \text{init}}$  of  $\mathcal{P}$ . Let  $i$  be the current round. In the first phase of the round the strategy  $C_{\mathcal{P}}^{\phi}$  is applied until an accepting state of the ASCC  $\mathcal{U}$  is reached. We use  $k_i$  to denote the number of steps we played the strategy  $C_{\mathcal{P}}^{\phi}$  in round  $i$ . Once the mission subgoal is fulfilled, the average subgoal becomes the current subgoal. In this phase, we play the strategy  $C_{\mathcal{P}}^V$  until the number of completed surveillance cycles in the second phase of the current round is  $l_i \geq \max\{j(\frac{1}{i}), i \cdot k_i\}$ .

*Theorem 1:* The strategy  $C_{\mathcal{P}}$  projects to a strategy  $C$  of  $\mathcal{T}$  that solves Problem 1.

*Proof:* From the fact that the ASCC  $\mathcal{U}$  is reachable from the initial state  $s_{\mathcal{P}init}$  and from the construction of  $C_{\mathcal{P}}^{\phi}$ , it follows that  $\mathcal{U}$  is reached from  $s_{\mathcal{P}init}$  in finite time. In every round  $i$  of the strategy  $C_{\mathcal{P}}$ , an accepting state is visited. Moreover, from Prop. 1 and the fact that  $l_i \geq \max\{j(\frac{1}{i}), i \cdot k_i\}$ , it can be shown that the average cumulative penalty per surveillance cycle incurred in the  $i$ -th round is at most  $V_{\mathcal{U}}^* + \frac{2}{i}$  with probability at least  $1 - \frac{1}{i}$ . Therefore, in the limit, the run induced by  $C_{\mathcal{P}}$  satisfies the LTL specification and reaches the optimal average cumulative penalty per surveillance cycle  $V_{\mathcal{U}}^*$  with probability 1. ■

Note that, in general, the strategy  $C_{\mathcal{P}}$  is not finite-memory. The reason is that in the modes of the finite-memory strategy we would need to store the number of steps spent so far in the first phase  $k_i$  and the number  $l_i$  of the surveillance cycles in the second phase of a given round. Since  $j(\frac{1}{i})$  is generally increasing with  $i$ , we would need infinitely many modes to be able to count the number of surveillance cycles in the second phase. However, if there exists a cycle  $\text{cyc}_{\mathcal{U}}^V$  of the SCC  $\mathcal{U}$  corresponding to  $V_{\mathcal{U}}^*$  that contains an accepting state, then the finite-memory strategy  $C_{\mathcal{P}}^V$  for the average subgoal maps to a strategy of  $\mathcal{T}$  solving Problem 1, which is therefore in the worst case finite-memory as well.

*Complexity:* The size of a BA for an LTL formula  $\phi$  is in the worst case  $2^{\mathcal{O}(|\phi|)}$ , where  $|\phi|$  is the size of  $\phi$  [17]. However, the actual size of the BA is in practice often quite small. The size of the product  $\mathcal{P}$  is  $\mathcal{O}(|S| \cdot 2^{\mathcal{O}(|\phi|)})$ . To compute the minimum weights  $w^*((s, q), (s', q'))$  between every two states of  $\mathcal{P}$  we use Floyd-Warshall algorithm taking  $\mathcal{O}(|S_{\mathcal{P}}|^3)$  time. Tarjan's algorithm [15] is used to compute the set  $\text{SCC}(\mathcal{P})$  in time  $\mathcal{O}(|S_{\mathcal{P}}| + |T_{\mathcal{P}}|)$ . The reduction of an ASCC  $\mathcal{U}$  can be computed in time  $\mathcal{O}(|S_{\mathcal{U}}| \cdot |T_{\mathcal{U}}|^2)$ . The Karp's algorithm [22] finds the optimal APPC value and corresponding cycle in time  $\mathcal{O}(|S_{\mathcal{U}}| \cdot |T_{\mathcal{U}}|)$ . The main pitfall of the algorithm is to compute the number  $j(\frac{1}{i})$  of surveillance cycles needed in the second phase of the current round  $i$  according to Prop. 1. Intuitively, we need to consider the finite run  $\sigma_{C_{\mathcal{P}}^V, k}$  induced by the strategy  $C_{\mathcal{P}}^V$  from the current state that contains  $k = 1$  surveillance cycles, and compute the sum of probabilities  $\Pr_{(s, q)}^{\mathcal{P}, C_{\mathcal{P}}}(\text{Cyl}((\sigma_{C_{\mathcal{P}}^V, k}, \tau)))$  for every  $\tau$  with the average cumulative penalty per surveillance cycle less or equal to  $V_{\mathcal{U}}^* + \frac{1}{i}$ . If the total probability is at least  $1 - \frac{1}{i}$ , we set  $j(\frac{1}{i}) = k$ , otherwise we increase  $k$  and repeat the process. For every  $k$ , there exist  $r^{|\sigma_{C_{\mathcal{P}}^V, k}|}$  sequences  $\tau$ . To partially overcome this issue, we compute the number  $j(\frac{1}{i})$  only at the point in time, when the number of surveillance cycles in the second phase of the current round  $i$  is  $i \cdot k_i$  and the average cumulative penalty in this round is still above  $V_{\mathcal{U}}^* + \frac{2}{i}$ . As the simulation results in Sec. VI show, this happens only rarely, if ever.

### C. Online control

The online algorithm locally improves the strategy  $C_{\mathcal{P}}$  according to the values of penalties observed from the current state and their simulation in the next  $h$  time units. The resulting strategy  $C_{\mathcal{P}}$  is again played in rounds. However, in each step of the strategy  $C_{\mathcal{P}}$ , we consider a finite set of finite

runs starting from the current state, choose one according to an optimization function, and apply its first transition.

Throughout the rest of the section we use the following notation. We use singletons such as  $u, u_i$  to denote the states of  $\mathcal{P}$ . Let  $\sigma_{\text{all}} \in \text{Run}_{\text{fin}}^{\mathcal{P}}(s_{\mathcal{P}init})$  denote the finite run executed by  $\mathcal{P}$  so far. Let  $i$  be the current round of the strategy  $C_{\mathcal{P}}$  and  $\sigma_i = u_{i,0} \dots u_{i,k}$  the finite run executed so far in this round, i.e.,  $u_{i,k}$  is the current state of  $\mathcal{P}$ . We use  $t_{i,0}, \dots, t_{i,k}$  to denote the points in time when the states  $u_{i,0}, \dots, u_{i,k}$  were visited, respectively.

The optimization function  $f: \text{Run}_{\text{fin}}^{\mathcal{P}}(u_{i,k}) \rightarrow [0, 1]$  assigns every finite run  $\sigma = u_0 \dots u_n$  starting from the current state a value  $f(\sigma)$  that is the expected average cumulative penalty per surveillance cycle that would be incurred in the round  $i$ , if the run  $\sigma$  was to be executed next, i.e.,

$$f(\sigma) = \frac{\sum_{j=0}^k g(u_{i,j}, t_{i,j}) + \sum_{j=1}^n g_{\text{sim}}(u_j, t_{i,k} + w_{\mathcal{P}}(\sigma^{(j)}))}{\sharp(\sigma_i \cdot \sigma(1) \dots \text{last}(\sigma))}, \quad (5)$$

where  $g_{\text{sim}}(u_j, t_{i,k} + w_{\mathcal{P}}(\sigma^{(j)}))$  is the simulated expected penalty incurred in the state  $u_j$  at the time of its visit. If the visit occurs within the next  $h$  time units and the state  $u_j$  is visible from the current state  $u_{i,k}$ , we simulate the penalty currently observed in  $u_j$  over  $w_{\mathcal{P}}(\sigma^{(j)})$  time units. Otherwise, we set the expected penalty to be  $g_{\text{exp}}(u_j)$ . The exact definition of  $w_{\mathcal{P}}(\sigma^{(j)})$  can be found in Tab. I.

For a set of states  $X \subseteq S_{\mathcal{P}}$ , we define a *shortening indicator function*  $I_X: T_{\mathcal{P}} \rightarrow \{0, 1\}$  such that for  $((s_1, q_1), (s_2, q_2)) \in T_{\mathcal{P}}$

$$I_X(((s_1, q_1), (s_2, q_2))) = \begin{cases} 1 & \text{if } w_{\mathcal{P}}^*((s_1, q_1), X) \\ & > w_{\mathcal{P}}^*((s_2, q_2), X), \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Intuitively, the indicator has value 1 if the transition leads strictly closer to the set  $X$ , and 0 otherwise.

In the first phase of every round, we locally improve the strategy  $C_{\mathcal{P}}^{\phi}$  computed in Sec. IV-B that aims to visit an accepting state of the chosen ASCC  $\mathcal{U}$ . In each step of the resulting strategy  $C_{\mathcal{P}}^{\phi}$ , we consider the set  $\text{Run}_{\phi}(u_{i,k})$  of all finite runs from the current state  $u_{i,k}$  that lead to an accepting state from the set  $F_{\mathcal{U}}$  with all transitions shortening in the indicator  $I_{F_{\mathcal{U}}}$  defined according to Eq. (6), i.e.,

$$\text{Run}_{\phi}(u_{i,k}) = \{\sigma \in \text{Run}_{\text{fin}}^{\mathcal{P}}(u_{i,k}) \mid \text{last}(\sigma) \in F_{\mathcal{U}}, \\ \forall 0 \leq j \leq |\sigma| - 1: I_{F_{\mathcal{U}}}((\sigma(j), \sigma(j+1))) = 1\}.$$

Let  $\sigma \in \text{Run}_{\phi}(u_{i,k})$  be the run that minimizes the optimization function  $f$  from Eq. (5). Then  $C_{\mathcal{P}}^{\phi}(\sigma_{\text{all}}) = \sigma(1)$ . Just like in the offline algorithm, the strategy  $C_{\mathcal{P}}^{\phi}$  is applied until a state from the set  $F_{\mathcal{U}}$  is visited.

In the second phase, we locally improve the strategy  $C_{\mathcal{P}}^V$  for the average subgoal computed in Sec. IV-B to obtain a strategy  $C_{\mathcal{P}}^V$ . However, the definition of the set of finite runs we choose from changes during the phase. At the beginning of the second phase of the current round  $i$ , we aim to reach the cycle  $\text{cyc}_{\mathcal{U}}^V = c_0 \dots c_m$  of the ASCC  $\mathcal{U}$  and we use the same idea that is used in the first phase above. To be specific, we define  $C_{\mathcal{P}}^V(\sigma_{\text{all}}) = \sigma(1)$ , where  $\sigma$  is the finite

$g_{\text{sim}}(u_j, t_{i,k} + w_{\mathcal{P}}(\sigma^{(j)})) = \begin{cases} g(u_j, t_{i,k}) + \frac{w_{\mathcal{P}}(\sigma^{(j)})}{r} & \text{if } u_j \in \text{Vis}(u_{i,k}), w_{\mathcal{P}}(\sigma^{(j)}) \leq h \text{ and } g(u_j, t_{i,k}) + \frac{w_{\mathcal{P}}(\sigma^{(j)})}{r} \leq 1, \\ \sum_{x=0}^r \text{pst}(\frac{x}{r}) + \text{pst}(1) & \text{if } u_j \in \text{Vis}(u_{i,k}), w_{\mathcal{P}}(\sigma^{(j)}) \leq h \text{ and } g(u_j, t_{i,k}) + \frac{w_{\mathcal{P}}(\sigma^{(j)})}{r} > 1, \\ g_{\text{exp}}(u_j) & \text{otherwise.} \end{cases}$
$\text{pst}(\frac{x}{r}) = \left( \sum_{y=0}^{z_1} \frac{(z_1-y+z_2+y(r+1))!}{(z_1-y)! \cdot (z_2+y(r+1))!} \cdot (1-p(u_j))^{(z_1-y)} \cdot (p(s))^{(z_2+y(r+1))} \right) \cdot (1-p(s)) \cdot \frac{x}{r}$ <p style="margin-top: 0;">if <math>z = w_{\mathcal{P}}(\sigma^{(j)}) - (1 - g(u_j, t_{i,k})) \cdot r - x - 1 \geq 0</math>, <math>z_1 = z \text{ div } (r+1)</math>, <math>z_2 = z \bmod (r+1)</math>; otherwise if <math>z &lt; 0</math>, <math>\text{pst}(\frac{x}{r}) = 0</math></p> $\text{pst}(1) = \sum_{y=0}^{z_3} \frac{(z_3-y+z_4+y(r+1))!}{(z_3-y)! \cdot (z_4+y(r+1))!} \cdot (1-p(u_j))^{(z_3-y)} \cdot (p(s))^{(z_4+y(r+1))} \cdot p(s)$ <p style="margin-top: 0;">where <math>z = w_{\mathcal{P}}(\sigma^{(j)}) - (1 - g(u_j, t_{i,k})) \cdot r - 1</math>, <math>z_3 = z \text{ div } (r+1)</math>, <math>z_4 = z \bmod (r+1)</math></p>

TABLE I: The function computing the simulated expected penalty incurred in a state  $u_j$  of the run  $\sigma$  at the time of its visit  $t_{i,k} + w_{\mathcal{P}}(\sigma^{(j)})$  if we are to apply the run  $\sigma$  from the current state  $u_{i,k}$ , *div* stands for integer division and *mod* for modulus.

run minimizing  $f$  from the set

$$\text{Run}_V(u_{i,k}) = \{\sigma \in \text{Run}_{\text{fin}}^{\mathcal{P}}(u_{i,k}) \mid \text{last}(\sigma) \in \text{cyc}_{\mathcal{U}}^V, \\ \forall 0 \leq j \leq |\sigma| - 1: I_{\text{cyc}_{\mathcal{U}}^V}((\sigma(j), \sigma(j+1))) = 1\}.$$

Once a state  $c_a \in \text{cyc}_{\mathcal{U}}^V$  of the cycle is reached, we continue as follows. Let  $c_b \in \text{cyc}_{\mathcal{U}}^V$  be the first state labeled with  $\pi_{\text{sur}}$  that is visited from  $c_a$  if we follow the cycle. Until we reach the state  $c_b$ , the optimal finite run  $\sigma$  is chosen from the set

$$\text{Run}_V(u_{i,k}) = \{\sigma \in \text{Run}_{\text{fin}}^{\mathcal{P}}(u_{i,k}) \mid \text{last}(\sigma) = c_b, \text{ and} \\ \forall 0 \leq j \leq |\sigma| - 1: I_{c_b}((\sigma(j), \sigma(j+1))) = 1 \text{ or} \\ |\sigma_{c_a \rightarrow u_{i,k}}| + |\sigma| \leq b - a + 2 \bmod (m+1)\},$$

where  $\sigma_{c_a \rightarrow u_{i,k}}$  is the finite run already executed in  $\mathcal{P}$  from the state  $c_a$  to the current state  $u_{i,k}$ . Intuitively, the set contains every finite run from the current state to the state  $c_b$  that either has all transitions shortening in  $I_{c_b}$  or the length of the finite run is such that if we were to perform the finite run, the length of the performed run from  $c_a$  to  $c_b$  would not be longer than following the cycle from  $c_a$  to  $c_b$ . When the state  $c_b$  is reached, we restart the above procedure with  $c_a = c_b$ . The strategy  $\mathbf{C}_{\mathcal{P}}^V$  is performed until  $l_i \geq \max\{j(\frac{1}{i}), i \cdot k_i\}$  surveillance cycles are completed in the second phase of the current round  $i$ , where  $k_i$  is the number of steps of the first phase and  $j$  is from Prop. 1. We can end the second phase sooner, specifically in any time when we complete a surveillance cycle and the average cumulative penalty per surveillance cycle incurred in the current round is less or equal to  $V_{\mathcal{U}}^* + \frac{2}{i}$ .

**Theorem 2:** The strategy  $\mathbf{C}_{\mathcal{P}}$  projects to a strategy  $\mathcal{C}$  of  $\mathcal{T}$  solving Problem 1.

*Proof:* First, we prove that Prop. 1 holds for the strategy  $\mathbf{C}_{\mathcal{P}}^V$  as well. This result follows directly from the facts below. The set of finite runs we choose from always contains a finite run induced by the strategy  $\mathbf{C}_{\mathcal{P}}^V$ . Once the cycle  $\text{cyc}_{\mathcal{U}}^V$  is reached, the system optimizes the finite run from one surveillance state of the cycle to the next, until it is reached after finite time. Finally, if the strategy  $\mathbf{C}_{\mathcal{P}}^V$  does not follow  $\mathbf{C}_{\mathcal{P}}^V$ , it is only because the chosen finite run provides lower expected average. The correctness of the strategy  $\mathbf{C}_{\mathcal{P}}$  is now proved analogously to the correctness of the strategy computed offline. ■

**Proposition 2:** The strategy  $\mathbf{C}_{\mathcal{P}}$  is with probability 1 expected to perform in the worst case as good as the strategy  $\mathcal{C}_{\mathcal{P}}$  computed offline. That means, if the average cumulative penalty per surveillance cycle incurred in the so far performed run of the system is lower than the optimal

APPC value  $V_{\mathcal{U}}^*$ , it will rise slower under the strategy  $\mathbf{C}_{\mathcal{P}}$  than under the strategy  $\mathcal{C}_{\mathcal{P}}$ . On the other hand, if the average cumulative penalty per surveillance cycle incurred in the so far performed run of the system is higher than the optimal APPC value  $V_{\mathcal{U}}^*$ , it is expected to decrease faster under the strategy  $\mathbf{C}_{\mathcal{P}}$  than under the strategy  $\mathcal{C}_{\mathcal{P}}$ .

*Proof:* Follows from the proof of Theorem 2. ■

**Complexity:** The cardinality of the set of finite runs  $\text{Run}_{\phi}(u_{i,k})$  grows exponentially with the minimum weight  $w_{\mathcal{P}}^*(u_{i,k}, F_{\mathcal{U}})$ . Analogously, the same holds for the set of finite runs  $\text{Run}_V(u_{i,k})$  and the set  $\text{cyc}_{\mathcal{U}}^*$  or one of its surveillance states. To simplify the computations and effectively use the algorithm in real time, one can use the following rule that was also applied in our implementation in Sec. VI. We put a threshold on the maximum weight of a finite run in  $\text{Run}_{\phi}(u_{i,k})$  and  $\text{Run}_V(u_{i,k})$ . In the second phase of a round, when on the optimal cycle, we optimize the finite run from the state  $c_a$  to the next surveillance state on the cycle  $c_b$ . However, if the weight of the fragment of the cycle from  $c_a$  to  $c_b$  is too high, we can first optimize the run to some intermediate state  $c'_b$ . Also, the complexity of one step of the strategy  $\mathbf{C}_{\mathcal{P}}$  grows exponentially with the user-defined planning horizon  $h$ . Hence,  $h$  should be chosen wisely. One should also keep in mind that the higher the planning horizon, the better local improvement.

## V. DISCUSSION

Every LTL formula  $\varphi$  over  $AP$  can be converted to a formula  $\phi$  of the form in Eq. (3) for which it holds that a run of the TS  $\mathcal{T}$  satisfies  $\phi$  if and only if it satisfies  $\varphi$ . The formula is  $\phi = \varphi \wedge \mathbf{GF} \pi_{\text{sur}}$  where  $\pi_{\text{sur}} \in L(s)$  for every  $s \in S$ . In that case, Problem 1 requires to minimize the expected average penalty incurred per stage.

The algorithms presented in Sec. IV can be used to correctly solve Problem 1 also for the systems with different penalty dynamics than the one defined in Sec. III. However, for every state we need to be able to compute the expected value of the penalty in the state, like in Eq. (2). For the online algorithm we also require that the dynamics of penalties allows to simulate them for a finite number of time units. More precisely, if we observe the penalty in a state  $s \in S$  in time  $t$ , we can compute the simulated expected value of the penalty in  $s$  in every following time unit, up to  $h$  time units, based only on the observed value.



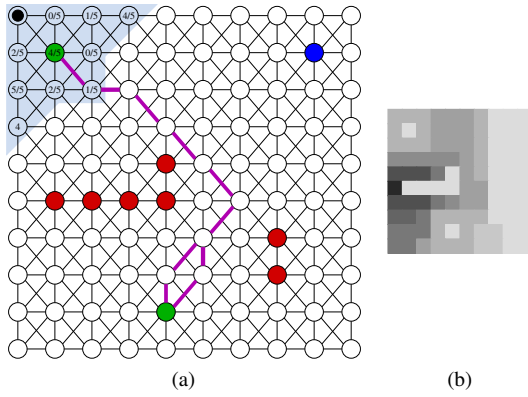


Fig. 2: (a) A TS modeling the robot (black dot) motion in a partitioned environment. Two stock locations are in green, a base is shown in blue, and unsafe locations are in red. There is a transition between vertically, horizontally or diagonally neighboring states. The weight of a horizontal and vertical transition is 2, for a diagonal transition it is 3. (b) The penalty probabilities in states. Darker shade indicates higher probability.

The online algorithm from Sec. IV-C is a heuristic. The sets of finite runs  $\text{Run}_\phi(u_{i,k})$ ,  $\text{Run}_V(u_{i,k})$  can be defined differently according to the properties of the actual problem. To guarantee the correctness of the strategy  $\mathbf{C}_P$ , the sets must satisfy the following conditions. There always exists a finite run in the set minimizing the optimization function  $f$  in Eq. (5). The definition of the set  $\text{Run}_\phi(u_{i,k})$  guarantees that an accepting state from  $F_U$  is visited after finite number of steps. The definition of  $\text{Run}_V(u_{i,k})$  also guarantees a visit of the cycle  $\text{cyc}_U^V$  in finite time and moreover, Prop. 1 holds for the resulting strategy  $\mathbf{C}_P^V$ .

## VI. CASE STUDY

We implemented the framework developed in this paper for a persistent surveillance robotics example in Java [25]. In this section, we report on the simulation results.

We consider a mobile robot in a grid-like partitioned environment modeled as a TS depicted in Fig. 2a. The robot transports packages between two stocks, marked green in Fig. 2a. The blue state marks the robot's base location. The penalties in states are defined by rate  $r = 5$  and penalty probability function in Fig. 2b. The visibility range  $v$  is 6. For example, in Fig. 2a the set  $\text{Vis}(s)$  of states visible from the current state  $s$ , with corresponding penalties, is depicted as the blue-shaded area. We set the planning horizon  $h = 9$ .

The mission for the robot is to transport packages between the two stocks (labeled with propositions  $a$ , and  $b$ , respectively) and infinitely many times return to the base (labeled with proposition  $c$ ). The red states in Fig. 2a are dangerous locations (labeled with  $u$ ) which are to be avoided. At the same time, we wish to minimize the cumulative penalty incurred during the transport of a package, *i.e.*, the surveillance property  $\pi_{\text{sur}}$  is true in both stock states. The corresponding LTL formula is

$$\mathbf{G}(a \Rightarrow \mathbf{X}(\neg a \mathbf{U} b)) \wedge \mathbf{G}(b \Rightarrow \mathbf{X}(\neg b \mathbf{U} a)) \wedge \mathbf{GF}c \wedge \mathbf{G}(\neg u) \wedge \mathbf{GF}\pi_{\text{sur}},$$

and the Büchi automaton has 10 states. The cycle providing the minimum expected average cumulative penalty per

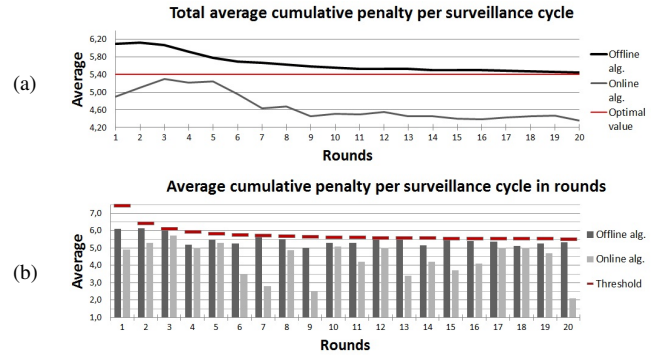


Fig. 3: (a) The average cumulative penalty per surveillance cycle incurred during the runs, shown at the end of each round. The red line marks the optimal APPC value. (b) The average cumulative penalty per surveillance cycle incurred in every round. The red bars indicate the threshold  $V_U^* + \frac{2}{i}$ .

surveillance cycle is depicted in magenta in Fig. 2a and the optimal APPC value is 5.4.

We ran both offline and online algorithm for multiple rounds starting from the base state. In Fig. 3 we report on the results for 20 rounds, for more results see [25]. As illustrated in Fig. 3a, the average cumulative penalty per surveillance cycle incurred in the run induced by the offline strategy is above the optimal value and converges to it fairly fast. For the run induced by the online strategy, the average is significantly below the minimum APPC value due to the local improvement based on local sensing. On the other hand, Fig. 3b shows the average cumulative penalty per surveillance cycle incurred in each round separately. The number of surveillance cycles performed in the second phase of every round  $i$  of the offline strategy was less than  $i \cdot k_i$ , *i.e.*, the second phase always ended due to the fact that the average incurred in the round was below the threshold  $V_U^* + \frac{2}{i}$ . The maximum number of surveillance cycles performed in the second phase of a round was 7. The same is true for the online strategy and the maximum number of surveillance cycles in the second phase of a round was 3. For both algorithms, the number of surveillance cycles in the second phase of a round does not evolve monotonically, rather randomly. Hence we conclude that in every round  $i$  we unlikely need to compute the value  $j(\frac{1}{i})$ .

## REFERENCES

- [1] C. Baier and J. Katoen, *Principles of model checking*. The MIT Press, 2008.
- [2] J. Barnat, L. Brim, M. Češka, and P. Ročkal, “DiVinE: Parallel Distributed Model Checker,” in *HiBi/PDMC’10*, 2010, pp. 4–7.
- [3] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, “PRISM: A Tool for Automatic Verification of Probabilistic Systems,” in *TACAS’06*, ser. LNCS, vol. 3920, 2006, pp. 441–444.
- [4] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, “NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking,” in *CAV’02*, ser. LNCS, vol. 2404, 2002.
- [5] S. L. Smith, J. Tumova, C. Belta, and D. Rus, “Optimal path planning for surveillance with temporal-logic constraints,” *I. J. Robotic Res.*, vol. 30, no. 14, pp. 1695–1708, 2011.
- [6] K. Chatterjee, L. Doyen, T. A. Henzinger, and J. F. Raskin, “Algorithms for omega-regular games with imperfect information,” in *CSL’06*, ser. LNCS, vol. 4207, 2006, pp. 287–302.
- [7] X. C. Ding, S. Smith, C. Belta, and D. Rus, “MDP Optimal Control under Temporal Logic Constraints,” in *CDC’11*, 2011, pp. 532–538.



- [8] P. Tabuada and G. Pappas, "Linear Time Logic Control of Discrete-Time Linear Systems," *Trans. on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.
- [9] E. A. Gol, M. Lazar, and C. Belta, "Language-Guided Controller Synthesis for Discrete-Time Linear Systems," in *HSCC'12*, 2012, pp. 95–104.
- [10] T. Wongpiromsarn, U. Topcu, and R. R. Murray, "Receding horizon temporal logic planning for dynamical systems," in *CDC'09*, 2009, pp. 5997–6004.
- [11] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *Trans. on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.
- [12] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta, "Temporal Logic Control of Discrete-Time Piecewise Affine Systems," *Trans. on Automatic Control*, vol. 57, pp. 1491–1504, 2012.
- [13] X. C. Ding, M. Lazar, and C. Belta, "Receding Horizon Temporal Logic Control for Finite Deterministic Systems," in *ACC'12*, 2012.
- [14] M. Svorenová, J. Tůmová, J. Barnat, and I. Černá, "Attraction-Based Receding Horizon Path Planning with Temporal Logic Constraints," in *CDC'12*, 2012, to appear.
- [15] R. Tarjan, "Depth-first search and linear graph algorithms," in *SWAT'71*, 1971, pp. 114–121.
- [16] M. Y. Vardi and P. Wolper, "Reasoning about Infinite Computations," *Information and Computation*, vol. 115, pp. 1–37, 1994.
- [17] P. Gastin and D. Oddoux, "Fast LTL to Büchi Automata Translation," in *CAV'01*, 2001, pp. 53–65.
- [18] F. Somenzi and R. Bloem, "Efficient Büchi Automata from LTL Formulae," in *CAV'02*, 2000, pp. 248–263.
- [19] P. Gastin and D. Oddoux. (2001) LTL 2 BA : fast translation from LTL formulae to Büchi automata. [Online]. Available: <http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/>
- [20] K. Chatterjee and L. Doyen, "Energy and Mean-Payoff Parity Markov Decision Processes," in *MFCS'11*, 2011, pp. 206–218.
- [21] R. Ash and C. Doléans-Dade, *Probability & Measure Theory*. Academic Press, 2000.
- [22] R. M. Karp, "A characterization of the minimum cycle mean in a digraph," *Discrete Mathematics*, vol. 23, no. 3, pp. 309–311, 1978.
- [23] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1994.
- [24] J. R. Norris, *Markov chains*, ser. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998.
- [25] M. Svorenova, I. Cerna, and C. Belta. (2012) Simulation of Optimal Receding Horizon Control for Finite Deterministic Systems with Temporal Logic Specification. [Online]. Available: [http://www.fi.muni.cz/~sim\\$175388/simulationLTLOptimalrh](http://www.fi.muni.cz/~sim$175388/simulationLTLOptimalrh)